http://www.ncic.ac.cn/~zjf

PreciseTracer

A precise, scalable and online

request tracing tool for multi-tier services

USER'S MANUAL

April 15th 2011

Revision Sheet

Release No.	Date	Revision Description
Rev. 1.0	15/4/2011	PreciseTracer v1.0 first packaging

USER'S MANUAL

TABLE OF CONTENTS

CONTENTS

1. Introduction 1

1.1. Targeted Audience 1

1.2. Structure of the document 1

1.3. Further Readings 1

2. PreciseTracer 3

2.1. Quick introduction 3

2.2. Available implementations 3

3. Getting started 4

3.1. Overview 4

3.2. Requirements 5

3.3. PreciseTracer distribution 5

3.4. PreciseTracer Architecture 7

3.5. Deploying PreciseTracer 8

3.5.1. RUBiS + Apache + JBoss + MySQL 8

3.5.2. IPlib inerpostion 8

3.5.3. SystemTap script 10

3.5.4. TCP_Tracer Component 10

3.5.5. Correlator Component 10

3.5.6. Analyzer Component 11

3.6. About WAP5 11

3.7. Running tests 12

3.7.1. Preparation 12

3.7.2. Offline mode 12

3.7.3. Online mode 12
3.7.4. Analyzing the results 13

4. Building your own PreciseTracer 14

4.1. Client emulators 14
4.2. Iplib interposition 14
4.3. SystemTap 14
4.4. TCP_Tracer Component 14
4.5. Correlator Component 14
4.6. Analyzer Component 15
4.7. new_WAP5 package 15
4.8. comparison package 15

5. References 16

Figures

Figure 1: Architecture of PreciseTracer
Figure 2: the Architecture of PreciseTracer

1. Introduction

PreciseTracer is a tracing and analyzing tool for multi-tier services of black boxes, which provides a convenient approach to achieving macro-level abstractions that enables debugging performance-in-the-large.

PreciseTracer v1.0 brings some simplicity in terms of installation, deployment and monitoring. With this version, we are offering an appliance with everything inside including the package of RUBiS used as a three-tier web service.

1.1. Targeted Audience

This document is targeting two types of audiences:

- Persons who just want to use PreciseTracer or WAP5 as a tracing or monitoring tool for their web services. This is for those who will directly use PreciseTracer v1.0 or WAP5 on similar three-tier platforms.

- Persons who would like to modify the sources to fit their particular needs. You could modify the source code of Precise Tracer or WAP5 to perform special tracing jobs or compare it with your approach or other tools.

1.2. Structure of the document

This document goes on the following route:

- A detailed introduction will be given in Section 2, for people who have never used RUBiS and PreciseTracer.

- How to install PreciseTracer version 1.0 or WAP5 in Section 3, for people who used the same platforms as shown in Figure. 1.

- How to build an appliance on your own needs in Section 4, for people who are going to modify some components in Figure. 1.

1.3. Further Readings

The following links give more in-depth details about technologies used in PreciseTracer v1.0.

- **RUBiS**: http://rubis.ow2.org/index.html
  http://rubis.ow2.org/doc/index.html

- **SystemTap**: http://sourceware.org/systemtap/documentation.html
  http://sourceware.org/systemtap/wiki

- **Library interposition**:

  http://developers.sun.com/solaris/articles/lib_interposers.html

- **PreciseTracer papers.**

Zhihong Zhang, Jianfeng Zhan, et al. Precise Request Tracing and Performance Debugging of Multi-tier Services of Black Boxes. Regular paper, The 39th IEEE/IFIP Dependable Systems and Networks (DSN 2009).

Bo Sang, Jianfeng Zhan, Gang Lu, et al. Precise, Scalable, and Online Request Tracing of Multi-tier Services of Black Boxes, Under Review (Major review) of IEEE Transaction on Parallel and Distributed Systems (TPDS).

- **WAP5 paper**

P. Reynolds, et al. WAP5: Black-box Performance Debugging for Wide-area Systems. In Proc. 15th WWW, 2006, pp.347-356.

2. PreciseTracer

2.1. Quick introduction

PreciseTracer is a tracing and monitoring tool for multi-tier services. Unlike other existing monitoring tools, it achieves high tracing accuracy, fast response, low overhead and scalability, doesn't needs source codes of applications, and has the functionality of online tracing and analyzing.

Our tests of PreciseTracer v1.0 are all based on RUBiS which is an auction sit benchmark using the typical three-tier model. Logs are collected on each server node by SystemTap probes interposed into a kernel module. Correlation and analyzation of the logs are deployed on another inner server node.

For further reading about PreciseTracer, please look at the following site: http://www.ncic.ac.cn/~zjf .

2.2. Available implementations

You may find available information and descriptions about older PreciseTracer versions at our home page . If newer version implemented, it will be appended.

If you find some bugs, please contact us via lugang@ncic.ac.cn.

If you successfully implement it on your own platform, please let us know.

If you have some novel idea, you might share with us.

3. Getting started

In this part, you will drive right into the configuration and running part, supposing you don't want to modify the provided PreciseTracer image. Figure. 1 will be called PreciseTracer v1.0 image reference.

3.1. Overview

Our experiment platform is based on RUBiSVA image reference in section 3.1 of rubisva_v1.0.pdf (http://rubis.ow2.org/download/rubisva_v1.0.pdf), which is a typical three-tier web application. Hence, it could be used on any Linux with some new kernel (more information in section 3.5). It offered the following architecture:

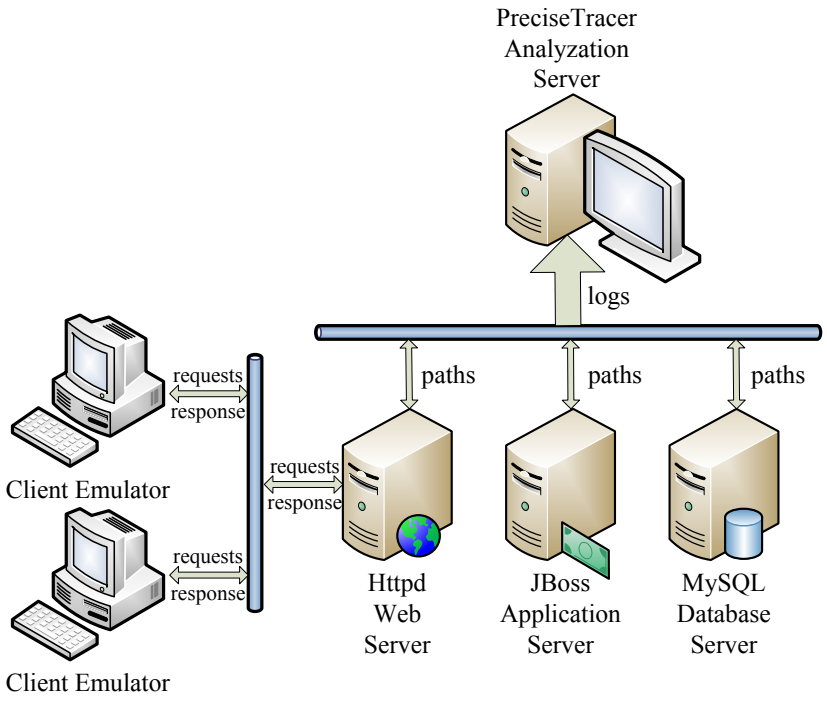

Figure 1 Architecture of PreciseTracer

- **Client Emulator(s)**: injecting the workload thanks to the RUBiS client (written in java) and collecting metric results. The emulator number can be changed if necessary.

- **Web Server**: handing HTTP requests from clients and able to load balance between multiple Application servers. We use Apache 2.2.16 as the front end.

- **Application Server**: serving client requests by running RUBiS server on JBoss. We use JBOSS 5.1.0.GA as the application server.

- **Database Server**: containing RUBiS DB. We use MySQL 5.1.49 as the back end.

- **Analyzation Server**: controlling the collecting and sleeping operations on the above three tracing nodes, copying logs from other sever nodes and analyzing the logs. We deploy the analyzer component on a common Linux server.

3.2. Requirements

The provided PreciseTracer v1.0 relies on Library interposition, SystemTap and RUBiS. In this part , we focus on how you can use what is provided in the PreciseTracer v1.0 package for deeper information you may go over the Building part in section 4. To do the following, you're suggested getting the root privileges for your Linux server.

RUBiS requirements: version 1.4.2 of RUBiS. For more software requirements, please look for http://rubis.ow2.org/doc/compile_and_run.html .

Three-tier web application servers:

Apache: 2.2.16 or newer

JBoss: 5.1.0.GA or newer

MySQL: 5.1.49 or newer

SystemTap: version 1.1-3.el5. Use the right same version as us unless you are familiar with differences of various versions. Make sure that the dependent packages has been installed.

IPlib interposition: available in the package

Analyzation server (also called management node): available in the package

3.3. PreciseTracer distribution

The PreciseTracer v1.0 distribution is inside pricisetracer_1.0.tar downloadable from the location as this document. Here is the structure of the precisetracer_1.0.tar:

RUBiS_client/
|_ rubis.properties
|_ …
SystemTap_script/
|_ hyb_temp
IPlib_src/
|_ Release/
|_ src/
|_ doc/
|_ include/
PreciseTracer_1.0/
|_ server_nodes_deployment/
|_ analyze_node_deployment/
|_ My_TraceAdapter/
|_ Release/

```
            |_ src/
        |_ My_Correlator/
            |_ Release/
            |_ src/
        |_ My_Analyzer/
            |_ MyPathAnalysis
            |_ Classification_PathPatterns
    new_WAP5/
        |_ Release/
        |_ src/
    comparison_package/
        |_ src/
```

Under the package, you will find the following files:

server_nodes_deployment/		
	run_gather_logs.sh	start tracing on the service nodes
	stop_gather_logs.sh	stop tracing
	deal_with_logs.sh	deploy local jobs on logs, including generating tuples from origin logs
	hyb_temp	SystemTap script used to generate SystemTap module
	main_process.c	generate *Gather*, communicate with management node
	send_logs_client.c	generate *Send_Message*, send message to management node
analyze_node_deployment		
	run_analysis.sh	start controlling server nodes
	stop_analysis.sh	stop managing
	analyze.sh	analyze the tracing tuples after get three log files by "scp", one from each server node
	Control_process.c	generate *Controler*, control server nodes by sending short messages
	Server_for_analysis.c	generate *Server*, call *analyze.sh* if get 3 log files, one for each server node
	compute_avg_duration.cpp	generate *Compute_avg_duration*, compute average response time of these paths
comparison_package		
	wap5_prepare_from_iplib.sh	generate IPlib correlating results and prepare wap5 logs from IPlib logs
	wap5_final.sh	do wap5 correlating and compare the path results with that of IPlib
	wap5_iplib_auto_all.sh	do what *wap5_prepare_from_iplib.sh* and *wap5_final.sh* do

	pt_prepare_tuples.sh	transform origin logs into tuples by My_TraceAdapter
	pt_correlate.sh	correlating tuples into paths (CAGs)
	pt_compare.sh	compare the path results with that of IPlib
	pt_autorun_all.sh	do what *pt_prepare_tuples.sh*, *pt_correlate.sh* and *pt_compare.sh* do
IPlib_src/doc		
	readconf.c	generate *readconf*, control the interposing of IPlib

3.4. PreciseTracer Architecture

PreciseTracer v1.0 has three main components: TCP_Tracer Component, Correlator Component and Analyzer Component. Client emulators are not integrated into PreciseTracer v1.0 as shown in Figure. 2, as it may be different for different appliances.

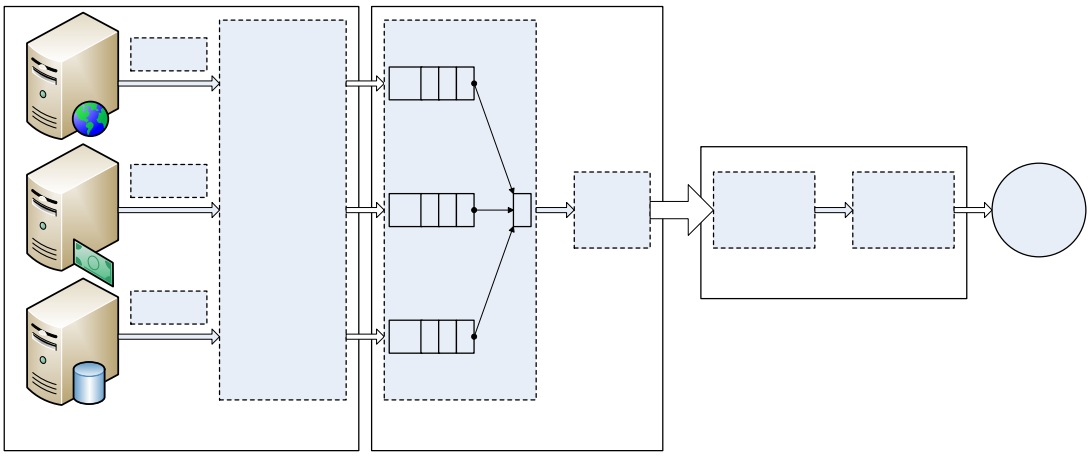

Figure 2 the Architecture of PreciseTracer

TCP_Tracer component uses SystemTap modules to collect logs which include specific process name in the context strings, such as httpd, java, mysql and so on. To get fully accurate logs and request paths for further comparison, we interpose IPlib into three server nodes besides SystemTap modules.

Correlator component and Analyzer component are on the analyzation node. Correlator component copies log tuples from three web server nodes and correlates them into paths (CAGs) by Ranker and Engine modules. Analyzer component conducts post processing like classifying and obtaining other statistics.

Ranker

Gather

…

Gather

3.5. Deploying PreciseTracer

3.5.1. RUBiS + Apache + JBoss + MySQL

To deploy RUBiS, you can look up http://rubis.ow2.org/doc/index.html and http://rubis.ow2.org/doc/install.html. The general process can be summarized to be following steps:

- Configuring Apache HTTP server
- Configuring Tomcat servlets server
- Configuring JBoss EJB server
- Setting up rubis.properties
- Compiling and running RUBiS

For Apache, you should modify the file *APACHE_ROOT/config/httpd.conf* when deploying RUBiS.

For JBoss, you should modify file *JBOSS_ROOT/server/default/deploy/mysql-ds.xml* when deploying RUBiS.

For MySQL, you should add user *rubis* when deploying RUBiS.

After deploying RUBiS, you should modify *rubis.properties* according to your tests. To learn how to configure the client emulator, please refer to http://rubis.ow2.org/doc/config.html#rubis_prop . At last, execute "make emulator" and RUBiS will get to work, emulating clients' visiting the auction site.

Don't go further unless you succeed in deploying RUBiS.

3.5.2. IPlib inerpostion

- disable IPv6

Disable IPv6 services on Apache node (It is unnecessary for other nodes).

For centos 5.4 or above, the following method can be useful:
add the following line into /etc/modprobe.conf:
    install ipv6 /bin/true
add the following lines into /etc/sysconfig/network:
    NETWORKING_IPV6=no
    IPV6NIIT=no
execute the following commands:
    service network restart
    reboot
check if IPv6 module has been removed by "lsmod | grep ipv6".

- build the library

  Path: Release/

execute the following commands:

```
make clean
make
make install
```

Default library path in Makefile is /usr/lib64, which is for 64-bit system. For 32-bit system, the library path will be /usr/lib.

- build the management

  Path: doc/

Generate management tool for IPlib:

```
gcc -O2 -o readconf readconf.c –ltrace
```

Note that sample.xml must be in the same path as readconf. Program "readconf" has three usages:

```
./readconf start
./readconf stop
./readconf status
```

Modify sample.xml according to your deployment. For example, if the IP of the Apache, JBoss and MySQL servers are 10.10.104.98, 10.10.104.97 and 10.10.104.93 respectively, the corresponding sample.xml will be like this:

```
<?xml version="1.0"?>
<conf>
    <Hosts nlayer="3" nhost="3" myid="0">
        <host id="0" layer="0" ip="10.10.104.98" proc="httpd"/>
        <host id="1" layer="1" ip="10.10.104.97" proc="java"/>
        <host id="2" layer="2" ip="10.10.104.93" proc="mysqld"/>
    </Hosts>
    <Log path="/var/log/rubis-tracing" />
</conf>
```

Note that the number of nodes begins from 0.

When you want to interpose IPlib, make sure the following line has been added into respective files.

```
export LD_PRELOAD = "/usr/lib64/libtrace.so"
```

For Apache node, at the beginning of APACHE_PATH/bin/envvars.
For JBoss node, at the beginning of JBOSS_PATH/bin/run.sh.
For MySQL node, at the beginning of MYSQL_PATH/bin/mysqld_safe.

Errors will occur if you start Apache, JBoss and MySQL without IPlib been interposed.

Create the log file in each node before starting tracing by IPlib:

```
touch /var/log/rubis-tracing
chmod 666 /var/log/rubis-tracing
```

3.5.3. SystemTap script

Path: TCP_Tracer_ROOT(the root path of TCP_Tracer Component)

SystemTap on our platform are of version 1.1-3.el5. If errors occur on different versions, you may probably check if the probes functions in the script are still proper.

Just execute the following commands to get the stap module on each server node.

```
hyb_temp –n APP_NAME
"CTRL+c"
copy stap**.ko from /tmp/stap**/ to TCP_Tracer_ROOT.
For Apache node, APP_NAME should be httpd;
For JBoss node, APP_NAME should be java;
For MySQL node, APP_NAME should be mysqld.
```

Make sure there are only one stap**.ko in the path of TCP_Tracer_ROOT.

3.5.4. TCP_Tracer Component

This component has three major functionalities: communicating with controlling service of Correlator Component, controlling the tracing and transforming logs into specific tuples.

Make sure the new SystemStap module is in the path of TCP_Tracer_ROOT.

Modify some configuration in file "run_gather_logs.sh" and "deal_with_logs.sh" in three server nodes like this:

```
ANALYSIS_NODE=10.10.104.100
LOCAL_NODE_IP=10.10.104.98      for Apache
LOCAL_NODE_IP=10.10.104.97      for JBoss
LOCAL_NODE_IP=10.10.104.91      for MySQL
```

Run script "run_gather_logs.sh" to start tracing the messages in each server node.

Run script "stop_gather_logs.sh" to stop SystemTap tracing.

3.5.5. Correlator Component

This component has three major functionalities: communicating with Correlator Component to control tracing tasks of the server nodes according to configuration of collecting and sleeping time, transmitting tuples from 3 sever nodes and correlating tuples into paths (CAGs).

Modify some configuration in file "run_analysis.sh" and "analyze.sh" in analyzation node like this:

```
HTTPD_NODE="10.10.104.98"
JBOSS_NODE="10.10.104.97"
MYSQL_NODE="10.10.104.93"
HTTPD_REMOTE_TUPLE_DIR="/opt/sangbo/httpd_tuples"
JBOSS_REMOTE_TUPLE_DIR="/opt/sangbo/jboss_tuples"
MYSQL_REMOTE_TUPLE_DIR="/opt/sangbo/mysql_tuples"
```

Modify the collecting_time and sleeping_time (in seconds) in the file of "run_analysis.sh" like this:

```
COLLECT_TIME_WINDOW=120
SLEEP_TIME_WINDOW=0
```

Run script "run_analysis.sh" to start the controlling service.

Run script "stop_analysis.sh" to stop the controlling service.

3.5.6. Analyzer Component

This component has only one functionality, which is analyzing the paths generated by Correlator Component. Its analyzing programs are integrated into the script "analyze.sh", as every correlating operation generates its paths, which will be removed right before next correlating. For offline mode, you can do the analyzing manually after correlating.

The analyzing operation in "analyze.sh" is as follows in our experiments:

```
./Classification_PathPatterns $PATHS_DIR/paths_$NUM/
$ERRORS_DIR/errors_$NUM/ result duration/duration_$NUM 10 >>debug
./Compute_avg_duration $PATHS_DIR/paths_$NUM/ duration
```

Codes in comparion_package shows how to compare PreciseTracer andWAP5 with IPlib(accuracy approach).

3.6. About WAP5

WAP5 is a back-box tracing approach for wide-area systems (*P. Reynolds, et al. WAP5: Black-box Performance Debugging for Wide-area Systems. In Proc. 15th WWW, 2006, pp.347-356.*). we find that PreciseTracer v1.0 achieves higher tracing accuracy and faster response time with respect to WAP5.

From introduction in Section 3.3, we can know how to get path results of WAP5. Hence, we can compare our approach PreciseTracer with it.

In the new_WAP5 directory, there are source codes of WAP5 mostly implemented in C++. For the origin paper of WAP5, please refer to "*P. Reynolds, et al. WAP5: Black-box Performance Debugging for Widearea Systems. In Proc. 15th WWW, 2006, pp.347-356.*".

3.7. Running tests

3.7.1. Preparation

Altering rubis.properties

major configuration in our tests:

```
workload_number_of_columns = 27
workload_number_of_rows = 29
workload_maximum_number_of_transitions = 1000
workload_number_of_items_per_page = 20
workload_use_tpcw_think_time = no
workload_up_ramp_time_in_ms = 60000
workload_up_ramp_slowdown_factor = 2
workload_session_run_time_in_ms = 300000
workload_down_ramp_time_in_ms = 60000
workload_down_ramp_slowdown_factor = 3
```

Getting IPlib interposed and ready to collecting

suppose you have correctly configured “sample.xml”
check modification in 3.5.1

```
./readconf start
./readconf status                (check the configuration)
```

Starting Apache, JBoss and MySQL

For both offline mode and online mode, configuration of “run_gather_lgs.sh” in Apache, JBoss and MySQL nodes are the same. There is only one difference between offline mode and online mode, that is different configuration of collecting time and sleeping time in “run_analysis.sh”.

```
./run_gather_lgs.sh              (run it in each server node)
```

3.7.2. Offline mode

For configuration (offline, 20, 100, 400, 1), set the collecting time to be 20(seconds) and sleeping time to be 0(seconds).

Run “./run_analysis.sh” to start serving.

Run “./stap_analysis.sh” after client emulator stops

3.7.3. Online mode

For configuration (online, 20, 100, 400, 10), set the collecting time to be 20(seconds) and sleeping time to be 100(seconds).

Run “./run_analysis.sh” to start serving.

Run “./stap_analysis.sh” after client emulator stops

3.7.4. Analyzing the results

Main analyzing jobs are done by Classification_PathPatterns and Compute_avg_duration.

Directory of the paths results: *my_path_results/*

Directory of duration: *duration/*

4. Building your own PreciseTracer

If you want to build your own PreciseTracer, this part will give some advices. If following introductions does not suffice for your approach, you may contact us via jfzhan@ncic.ac.cn.

4.1. Client emulators

If you use emulators other than RUBiS, you'd better make sure they are in blocking mode. That is because the Correlator Component has only been verified efficient when requests are sent in blocking ways. For unblocking requests the feasibility is unclear, which we would figure out in the future. Maybe you can try some experiments and share us with the results.

4.2. Iplib interposition

For deployment with different amount of server nodes, just modify sample.xml and make sure the command "export …" has been added into relevant bash files.

```
export LD_PRELOAD = "/usr/lib64/libtrace.so"
For Apache node, at the beginning of APACHE_PATH/bin/envvars.
For JBoss node, at the beginning of JBOSS_PATH/bin/run.sh.
For MySQL node, at the beginning of MYSQL_PATH/bin/mysqld_safe.
```

Don't forget to modify *sample.xml* on your needs.

4.3. SystemTap

Write SystemTap script on your own needs or change APP_NAME into your application name. For example, use "hyb_temp –n dhcpd" to monitor the process "dhcpd".

4.4. TCP_Tracer Component

In this component, you may only modify the communication program with analyzing node. Section 3.3 has shown where the source codes are.

4.5. Correlator Component

You might not modify our correlator component except the communication part, which has been introduced in Section 3.3.

4.6. Analyzer Component

If you want to do more analyzing, first you must be familiar with our codes. After that, you may get to know how the CAGs are constructed. Knowing enough properties of the paths, you can start to write your own analyzing codes.

4.7. new_WAP5 package

The new_WAP5 package is just used for comparing with PreciseTracer v1.0. You may change the probability limitations in new_WAP5 for your own experiments.

4.8. comparison package

Like package new_WAP5, this package is no need to modify unless you have special needs.

5. References

http://www.ncic.ac.cn/~zjf/index.htm

http://rubis.ow2.org/index.html

http://rubis.ow2.org/doc/index.html

http://sourceware.org/systemtap/documentation.html

http://sourceware.org/systemtap/wiki

http://developers.sun.com/solaris/articles/lib_interposers.html

http://rubis.ow2.org/download/rubisva_v1.0.pdf

http://rubis.ow2.org/doc/compile_and_run.html

Z. Zhang, Jianfeng Zhan, et al. Precise Request Tracing and Performance Debugging of Multi-tier Servcies of Black Boxes. In Proc. DSN'09, 2009, pp.337-346.

Bo Sang, Jianfeng Zhan, Gang Lu, et al. Precise, Scalable, and Online Request Tracing of Multi-tier Services of Black Boxes, Under Review (Major review) of IEEE Transaction on Parallel and Distributed Systems (TPDS).

P. Reynolds, et al. WAP5: Black-box Performance Debugging for Widearea Systems. In Proc. 15th WWW, 2006, pp.347-356.